9. Каневская Р.Д. Математическое моделирование гидродинамических процессов разработки месторождений углеводородов. – М. – Ижевск: Институт компьютерных исследований, 2003. – 127 с.

10. Пантелеев А.В., Летова Т.А. Методы оптимизации в примерах и задачах. – М.: Высшая школа, 2002. – 544 с.

Сергеев Виктор Леонидович
Д-р техн. наук, профессор кафедры автоматизированных систем управления ТУСУРа
Телефон: (3822) 41 34 54
Эл. почта: svl@mail.tomsknet.ru

Севостьянов Дмитрий Владимирович
Аспирант кафедры автоматизированных систем управления ТУСУРа
Эл. почта: sevostjanov@aurigma.com

D.V. Sevostjanov, V.L. Sergeev
Identification and prognosis of well deliverability in the process of functioning on the base of the method of integrated models

This work represents and analyzes the solving method of tasks of identification of seam pressure and production of wells in condition of normal operation. The method is based on present-day principles of systems analysis of mathematical models connected to physical processes of oil-and-gas production and data of complex well surveys.

УДК 519.7

Б.А. Соловьев, В.Т. Калайда

Моделирование работы распределенной системы видеонаблюдения

В работе рассматривается алгоритм анализа нагрузки распределенной системы видеонаблюдения, приводится оценка нагрузки методом имитационного моделирования и дается интерпретация полученных результатов моделирования.

При разработке современных программных систем важную роль играет задача интеграции отдельных элементов и целых подсистем в единую систему. Для обеспечения совместимости при интегрировании, как правило, используется одна из технологий компонентного программирования. При этом правила организации взаимодействия элементов реализуются в виде заранее оговоренных интерфейсов компонентов, представляющих собой элементы разрабатываемой системы [1]. Для каждой функции элемента разрабатывается специальный набор интерфейсов, а элемент, реализующий эту функцию, обязан предоставлять оговоренный интерфейс. Эта технология приводит к громоздким надстройкам над основным кодом при интеграции подсистем [2]. Нами предлагается новая схема организации взаимодействия элементов распределенных систем, реализованная в технологии «Базис».

Основным элементом технологии «Базис» является прикладной объект. Он реализует какую-либо функцию распределенной системы, принимает данные из внешней информационной среды и/или передает сгенерированные в нем данные в эту информационную среду. В зависимости от назначения объекта он может принадлежать к одному из нижеперечисленных классов:

- объект-приемник — объект, который имеет один или более входов, но не имеет выходов;
- объект-генератор — не принимает никаких данных из информационной среды, но имеет один или более выходов;
- комплексный объект — объединяет в себе объект-приемник и генератор и имеет как входы, так и выходы.

Объекты объединяются в подсистемы. Объекты и связи между ними, а также подсистемы целиком могут появляться и исчезать, останавливаться и начинать работу в процессе функционирования всех подсистем.

В процессе функционирования «Базис» работает только с прикладными объектами и так называемыми объектами-значениями. Объект-значение представляет собой поток данных конкретного выхода конкретного объекта, генерируя выходные данные, прикладной объект инициирует изменение значения, сопоставленного с его выходом. Таким образом, объекты-приемники получают состояние объекта-значения, а не поток с выхода объекта.

Построенная по такой технологии система может содержать достаточно сложные структуры связей элементов внутри себя, в том числе прямые и опосредованные обратные связи. Большое количество данных, циркулирующих в системе, приводит к большим нагрузкам как на локальную систему каждой из шин объектов, так и на сеть, в которой «Базис» функционирует. В этой связи возникает проблема управления интенсивностью потоков данных. У каждого поставщика данных «Базис», которыми являются объекты-генераторы, может быть несколько приемников. Каждый из приемников может работать на разных скоростях приема. Таким образом, задача, стоящая перед «Базис», разбивается на две подзадачи: не допустить перегрузку локальной системы, на которой работает очень медленный и тяжеловесный, с точки зрения использования ресурсов, прикладной объект, и синхронизировать частоту генерации данных с частотой приема этих данных самого требовательного приемника.

Задача управления скоростью также усложняется вследствие того, что скорости генерации не могут принимать любые значения. Это обусловлено внутренним устройством каждого генератора. Например, объект-генератор, работающий с камерой и поставляющий в информационную среду изображения, может поставлять данные только с частотой 5, 15, 25 или 30 кадров в секунду, что зависит от устройства, с которым объект работает.

Наличие в системе обратных связей может превратить процесс настройки скоростей в бесконечный. На рис. 1 показаны две наиболее распространенные разновидности связей, которые могут привести к «зависанию» процесса настройки. Здесь показаны упрощенные варианты этих связей, но в действительности они могут оказаться не столь очевидными.

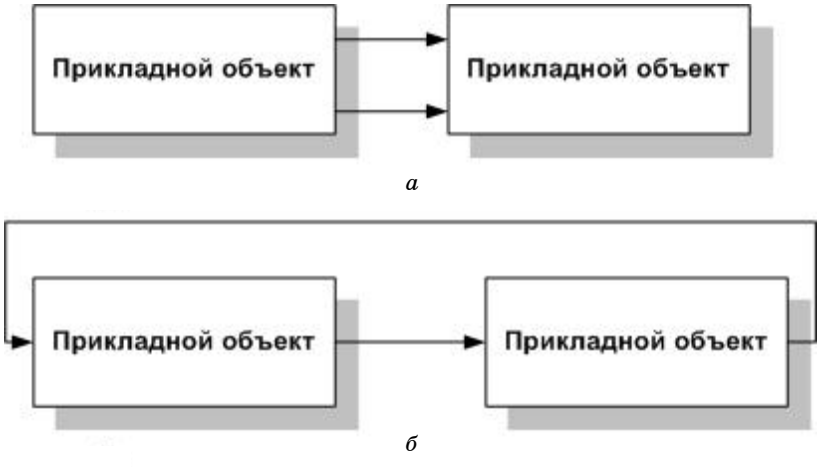

Рис. 1 — Варианты обратной связи при настройке скоростей потоков

В первом варианте (рис. 1,*а*) два выхода объекта-генератора подключаются к двум входам объекта-приемника. При изменении частоты генерации первого потока данных изменяется состояние первого входа приемника, что может повлечь за собой изменение состояния второго входа, потому что связь их состояний может быть обусловлена внутренним устройством приемника. Изменение второго входа приемника влечет за собой изменение состояния второго выхода генератора, потому что поток с прежними параметрами может не удовлетворять требованиям нового состояния второго входа. А это в свою очередь может повлечь изменение первого выхода генератора, и так далее до бесконечности.

Во избежание описанного конфликта введем ряд ограничений:

- инициация изменения состояния выхода одного из объектов может исходить от системы, пользователя или объекта, чей вход принимает данные рассматриваемого объекта;
- изменение состояния входа объекта может инициироваться только тем объектом, которому данный вход принадлежит;
- любой объект может изменить только состояние выхода объекта, подключенного к его входу.

Применим механизм ограничения трафика, называемый «Корзина» [2]. Корзина располагается на входе объекта-приемника и ограничивает входящий трафик. Если объект тратит много ресурсов системы на обработку входящих данных, а поток данных слишком большой, то «Корзина» передает данные на вход с той частотой, с которой приемник может их обрабатывать, не приводя систему к перегрузкам. При этом часть пакетов данных может теряться.

Ситуации, возникающие во втором случае (рис. 1,*б*), можно проиллюстрировать двумя примерами. Пусть прикладной объект, расположенный слева на рис. 1,*б*, может генерировать данные с частотами 1,3 и 5 пакетов в секунду, а объект, расположенный справа, — 2, 4 и 6 пакетов в секунду. Обозначим состояние выхода левого объекта $LO(x)$, где x — частота генерации данных на его выходе, его вход обозначим $LI(i0, i1)$, где $i0$ — частота входящих в корзину пакетов, $i1$ — частота пакетов, пропущенных «Корзиной» на вход. Соответственно состояния входа и выхода правого объекта — $RO(x)$ и $RI(i0, i1)$. Пусть в начальный момент времени мы имеем набор состояний $\{LI(4,5), LO(5), RI(5,4), RO(4)\}$ — так было задано в момент установки данных компонентов. Несмотря на то, что выход правого объекта не удовлетворяет потребности входа левого, такая ситуация вполне возможна. Положим, по одной из причин приходит запрос изменить состояние выхода правого объекта на $RO(2)$. Тогда

$$RO(2) \longrightarrow RI(5,2),$$
$$RO(2) \longrightarrow LI(2,5),$$
$$RI(5,2) \longrightarrow LO(3),$$
$$LO(3) \longrightarrow RI(3,2),$$
$$LO(3) \longrightarrow LI(2,3),$$
$$LI(2,3) \longrightarrow RO(4),$$
$$RO(4) \longrightarrow RI(3,4),$$
$$RO(4) \longrightarrow LI(4,3) \text{ и т.д.}$$

Сперва мы устанавливаем состояние выхода в требуемое и изменяем состояние нашего входа, затем изменяем состояние входа потребителя. Изменение состояния входа влечет за собой изменение состояния выхода левого объекта в $LO(3)$, так как он может принимать значения только $LO(1)$, $LO(3)$ или $LO(5)$. Продолжая эту цепочку, мы придем к тому, что состояние выхода правого объекта снова изменится на 4, а если продолжить цепочку далее, то процесс остановится при состоянии системы $\{LI(6,5), LO(5), RI(5,6), RO(6)\}$. Этот пример показывает, что в результате прохождения петли мы не только не получили желаемого результата, но настройка привела к неожиданным изменениям всей системы.

Второй пример заключается в том, что возможные частоты генерации у левого объекта лежат в области всех положительных четных чисел, а у правого — положительных нечетных. В этом случае процесс никогда не остановится.

Для моделирования опишем процесс настройки скоростей потоков в терминах теории взаимодействующих последовательных процессов. Процесс настройки состоит из следующих событий:

- z^p — активизация настройки скоростей для объекта p;
- a_o^p — изменение скорости выхода o объекта p;
- b_o^p — передача параметров выхода o объекта i всем его потребителям;
- c_i^p — запуск процесса настройки скорости выхода, подключенного к входу i объекта p;
- d_i^p — изменение скорости i-го входа p;
- e_i^p — изменить настройки корзины i-го входа объекта p,

и состоит из следующих процессов:

- O^p — основной процесс объекта p;
- A^p — процесс настройки скорости выхода объекта p;
- D_i^p — процесс настройки скорости входа i объекта p;
- E_o^p — процесс изменения параметров корзин всех входов, подключенных к выходу o объекта p.

Описание процесса настройки скоростей потоков данных объекта p выглядит следующим образом:

$$O^p = (z^p \to A^p); \quad A^p = \left(a_o^p \to E_o^p \middle| d_i^p \to D_i^p \middle| e_i^p \to A^p\right)^{\backprime};$$

$$E_o^p = (b_o^p \to A^p); \quad D_i^p = \left(e_i^p \to A^p\right)^{\backprime}.$$

Для устранения перечисленных выше возможных конфликтов, а также исключения параллельных запросов на настройку скоростей других объектов, протекание которых может привести к непредвиденным последствиям, вводим понятие сеанса настройки. В рамках одного сеанса настройки происходят изменения, которые вызывает начальный запрос, все остальные запросы в это время ставятся в очередь, а по окончании текущего сеанса настройки из очереди выбирается следующий запрос и для него создается новый сеанс. Все объекты, измененные во время сеанса настройки, помещаются в список, и при попытке изменения объекта из списка происходит отказ. Таким образом, «Базис» гарантирует, что инициированное пользователем или системой изменение вступит в силу и не приведет к зацикливанию.

На рис. 2 показана сеть Петри, иллюстрирующая модель алгоритма подстройки скоростей потоков, реализованного в «Базис» [3]. Событиями для этого алгоритма являются:

- инициализация системы;
- приход запроса на изменение скорости какого-либо выхода;
- определение того, что запрос не принадлежит текущему сеансу;
- создание нового сеанса;
- время ожидания запросов для текущего сеанса истекло;
- сеанс завершен;
- определение того, что запрос принадлежит текущему сеансу;
- новое значение скорости генерации потока максимально среди скоростей приема этого потока;
- новое значение скорости генерации потока не максимально среди скоростей приема этого потока;
- состояние входа изменялось во время текущего сеанса;
- состояние входа не изменялось во время текущего сеанса;
- новое значение скорости генерации передано объекту-генератору.

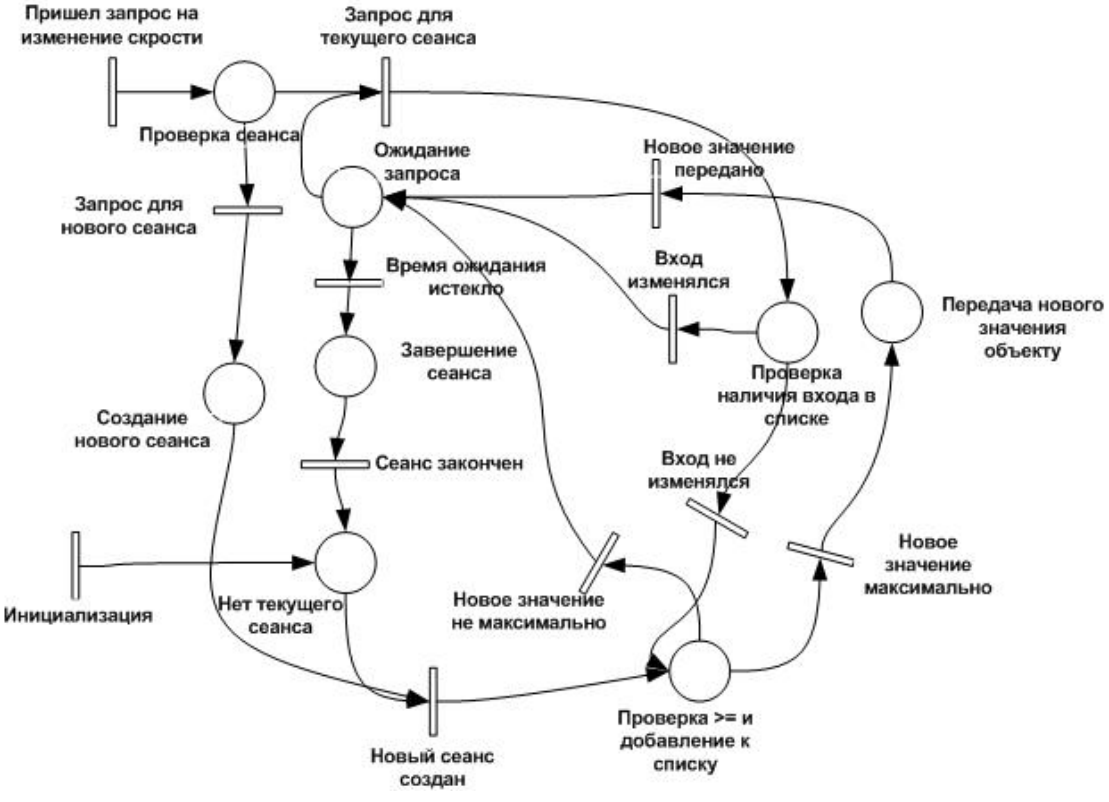

Рис. 2 — Сеть Петри, описывающая алгоритм подстройки скоростей потоков

Условиями для алгоритма являются:

- проверка принадлежности изменения текущему сеансу;
- создание нового сеанса;
- ожидание запроса для текущего сеанса;
- завершение сеанса;
- отсутствие текущего сеанса;
- проверка того, изменялось ли состояние входа во время текущего сеанса;
- проверка максимальности запрошенного значения скорости среди скоростей приема для каждого приемника и внесение выхода в список изменений независимо от того, является ли требуемое значение максимальным;
- передача запрошенного значения скорости объекту-генератору для принятия.

Фишки представляют собой запросы на изменение скорости. В начальный момент времени функционирования происходит инициализация системы, которая не повторяется больше никогда. Эта операция помещает одну фишку в позицию «нет текущего сеанса», что разрешает системе создать новый сеанс при поступлении первого запроса. Затем фишки появляются только в результате прихода нового запроса и уничтожаются в начале обработки запроса.

Для оценки пропускных способностей и допустимых нагрузок на локальные системы и на сеть, в которой функционирует «Базис,» целесообразно прибегнуть к технологии имитационного моделирования. Для этого была составлена программа моделирования в среде GPSS World [4]. При моделировании учитывались следующие параметры:

- среднее время генерации пакета данных;
- дисперсия времени генерации;
- длина генерируемых пакетов, которая сказывается на скорости передачи пакетов по сети;
- оценка количества машинных операций, требуемых для генерации одного пакета установленного типа;
- среднее время обработки пакета объектом-приемником;
- дисперсия времени обработки;
- оценка количества машинных операций, требуемая для обработки пакета заданного типа;
- пропускная способность сети;
- производительность компьютеров, задействованных в работе системы.

Среднее время обработки или генерации напрямую зависит от количества машинных операций, но использование обоих параметров необходимо учитывать, потому что в процессе генерации могут происходить задержки, не влияющие на загруженность системы, то есть время этих задержек процессор не использует. Это может быть связано с ожиданием данных от устройства. При моделировании были взяты параметры, характерные для функционирования системы в наихудших условиях. В модели учитывались пять типов сигналов, характерных для систем видеонаблюдения и контроля доступа:

- изображение с размером пакета 230454 байт;
- текстовое сообщение с размером пакета 255 байт;
- ключевые данные считывателей типа клавиатуры с пакетом длиной 100 байт;
- звуковой сигнал с размером пакета 64 байта;
- неформализованные пакеты данных длиной не более 1024 байт.

Среди четырех компьютеров были размещены шестнадцать генераторов различных типов сигналов и десять приемников соответствующих сигналов. Приемники принимали и обрабатывали сигналы соответствующего им типа от всех генераторов данного сигнала. То есть каждый генератор соединялся со всеми приемниками сигнала заданного типа. В таблице приведено распределение генераторов и приемников по компьютерам.

Оценка количества операций для генерации пакета происходила следующим образом. Программа генерировала сигналы с заданной частотой в течение пяти минут, при этом в журнал производительности записывался коэффициент утилизации процессора программой-генератором каждые 500 мс. В результате рассчитывалось время, в течение которого процессор занимала программа-генератор из пяти отведенных ей минут, и с использованием показателя производительности компьютера (количество операций в секунду) подсчитывалось примерное количество операций, требуемых для генерации данного типа сигнала. Оценка количества операций для обработки сигнала происходила аналогичным методом.

Расположение элементов моделируемой системы по компьютерам

Генераторы	Приемники
Компьютер № 1	
изображений ключей звука неформализованных данных	изображений неформализованных данных звука
Компьютер № 2	
изображений текста звука неформализованных данных	изображений текста неформализованных данных
Компьютер № 3	
изображений текста ключей неформализованных данных	ключей текста
Компьютер № 4	
изображений текста ключей звука	звука ключей

Для моделирования производительность компьютеров была принята равной 1 млрд операций в секунду, а пропускная способность сети — 10 Мбит. Моделирование происходило в течение пяти минут. При этом общее число сгенерированных пакетов составило 5760, из них изображений — 1376, текстовых сообщений — 1065, ключевых данных — 1106, звуковых пакетов — 1101 и неформализованных пакетов — 1112. Средняя загрузка компьютеров составила 27, 31, 22 и 98 %. Загрузка сети составила 29 %. На момент окончания моделирования в очередях необработанными остались 774 изображения, 507 пакетов с ключевыми данными и 511 пакетов неформализованных данных. Таким образом, данная схема с сознательно ухудшенными параметрами обеспечивает разумную производительность, что свидетельствует об ее эффективности и правильности решений, заложенных в структуру «Базис».

На базе описанной технологии в лаборатории межведомственной научно-исследовательской группы Института оптики атмосферы СО РАН была реализована распределенная система видеонаблюдения. В системе задействованы две машины и пятнадцать объектов семи различных типов. Реализованная система снимает видеопотоки с трех камер, подключенных через коммутатор видеоканалов к одной из машин системы, визуализирует их содержимое на дисплеи двух машин, а также проводит анализ наличия движущихся или оставленных предметов в зоне видимости камер и ведет видеоархив на три камеры. В процессе функционирования описанной системы в течение трех суток было сгенерировано 518400 пакетов видеоданных, из них отправлено через сеть 64800 пакетов данных. Средняя загруженность компьютера с процессором Pentium 2.4 MHz, отвечающего за получение видео, датчики движения, протоколирование и захват лица, составила 38 %, а компьютера Celeron 2 MHz, визуализирующего данные с камер, — 4 %. За это время подсистемой распределения нагрузки было инициировано 3 сеанса подстройки частот генерации.

Литература

1. Цимбал А.А., Аншина М.Л. Технологии создания распределенных систем. – СПб.: Питер, 2003. – 576 с.

2. Таненбаум Э., ван Стен М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877 с.

3. Питерсон Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 264 с.

4. Шрайбер Т. Дж. Моделирование на GPSS. – М.: Машиностроение, 1980. – 592 с.